Windows Programming

Visual C++ MFC Programming

Lecture 08

김예진
Dept. of Game Software

Notices

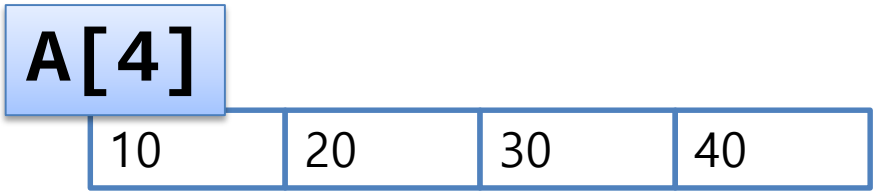
- 03/07: 502 → 501 등록 이동
- 03/21: HW 1 (Due: 03/28)
- 04/09: HW 2 (Due: 04/16)
- 04/25: Midterm (실습, ~75 min, 강의록 1~7)

Plan

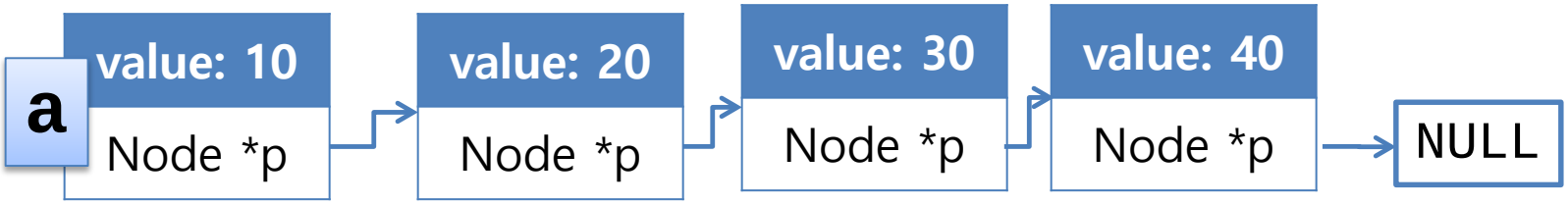
- Linked List
  - Template과 STL
  - CList

Linked List

- 많은 data를 저정하는 방법
  - Array 이용



  - Linked List 이용



Linked List

- 구조체와 pointer를 이용한 Linked List 정의
  - 구조체 내에 pointer를 member로 가질 수 있음

```
struct List
{
     int id;
     int *p;
};
```

  - 구조체를 가리키는 pointer를 member로 가질 수 있음

```
struct List
{
     int id;
     List *p;
};
```

Linked List

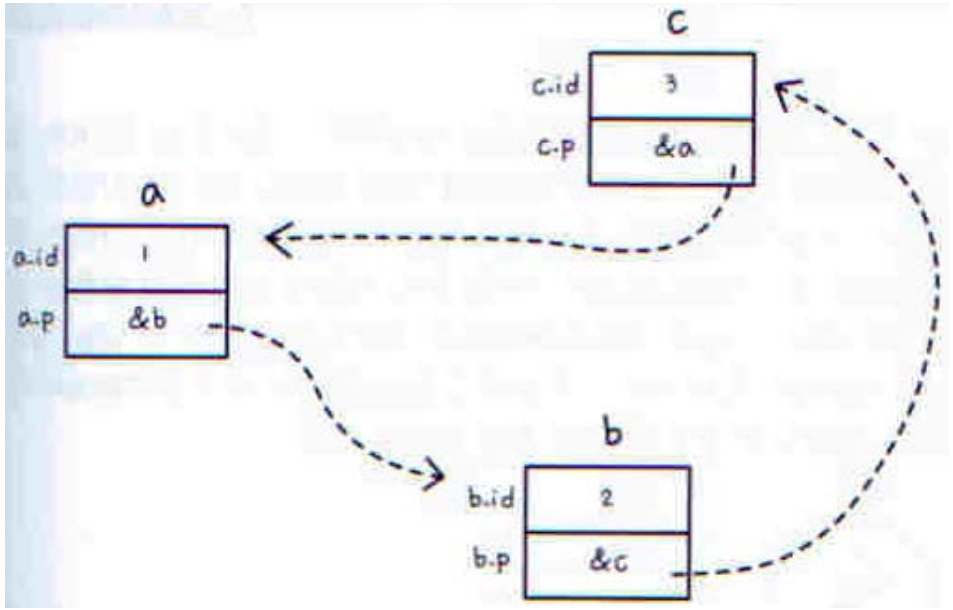
- 구조체와 pointer를 이용한 Linked List 사용 (1/2)
  - 구조체를 가리키는 member pointer 사용 예

```
#include <iostream>
using namespace std;

struct List
{
        int id;
        List *p;
};

void main ()
{
        List item;
        item.id = 20;
        item.p = &item;

        cout << "item.id: " << item.id << "\n";
        cout << "item.p->id: " << item.p->id << "\n";
        cout << "item.p->p->id: " << item.p->p->id << "\n";
}
```

Linked List

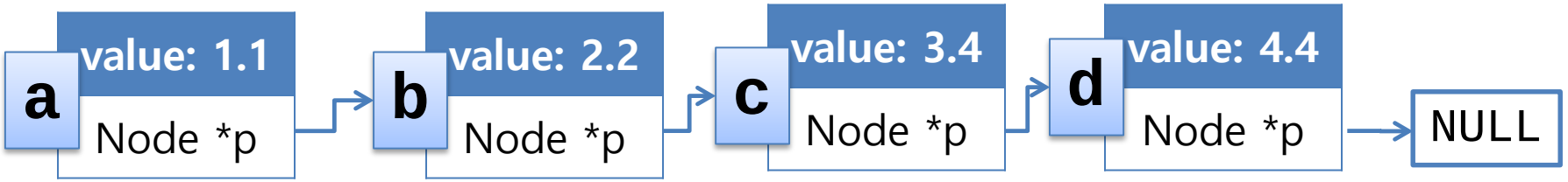
- 구조체와 pointer를 이용한 Linked List 사용 (2/2)
  - 구조체를 가리키는 member pointer 사용 예

```
#include <iostream>
using namespace std;

struct List
{
        int id;
        List *p;
};

void main ()
{
        List a, b, c;
        a.id = 1;  a.p = &b;
        b.id = 2;  b.p = &c;
        c.id = 3;  c.p = &a;

        cout << "a.id: " << a.id << "\n";
        cout << "b.id: " << a.p->id<< "\n";
        cout << "c.id: " << a.p->p->id<< "\n";
}
```



Linked List

- 사용 예: 다음과 같이 Node 변수 4개를 linked list로 연결 후 각 변수의 값을 출력



  - a부터 시작해 반복문(while) 을 통해 모든 변수의 value 값을 출력
  - Node는 구조체와 pointer로 정의

```
struct Node
{
    float value;
    Node *p;
};
```

Template과 STL

- **템플릿(Template): tem·plate** n.
  1. 본뜨는 공구(工具), 형판(型板)
  2. 【건축】 보받이, 도리받이
  3. 조선대(造船臺)의 쐐기;(반)투명의 피복지(彼覆紙)
  4. 【생화학】 (핵산의) 주형(鑄型)
  5. 【컴퓨터】 보기판, 템플릿《키보드 위에 놓고 각 키에 할당된 명령의 내용을 보이는 시트》

Template과 STL

- 일반성이 필요한 예
  - 두 정수 중 큰 수를 알려주는 max 함수

```
int max(int a, int b)
{
     if (a>b) return a;
     else return b;
}
```

- 두 double 간의 비교가 필요하다면? 오버로딩! OVERLOADING!
- 두 char 간의 비교가 필요하다면? 오버로딩! OVERLOADING!
- 두 complex 간의 비교가 필요하다면? 오버로딩! OVERLOADING!

한 번에 해결할 수 없을까?

➔ **Template 사용**

Template과 STL

- Template 사용

Template과 STL

- Template의 활용 방법
  - Template class + Template function
  - Smart pointer 활용 예

```
class AutoArray
{
public:
      AutoArray(int *ptr)
      {     _ptr = ptr; }
      ~AutoArray()
      {     delete [] _ptr; }
      int& operator[] (int index)
      {     return _ptr[index];   }
private:
      int *_ptr;
};

int main()
{
      AutoArray arr( new int[100] );
      arr[20] = 30;

      return 0;
}
```

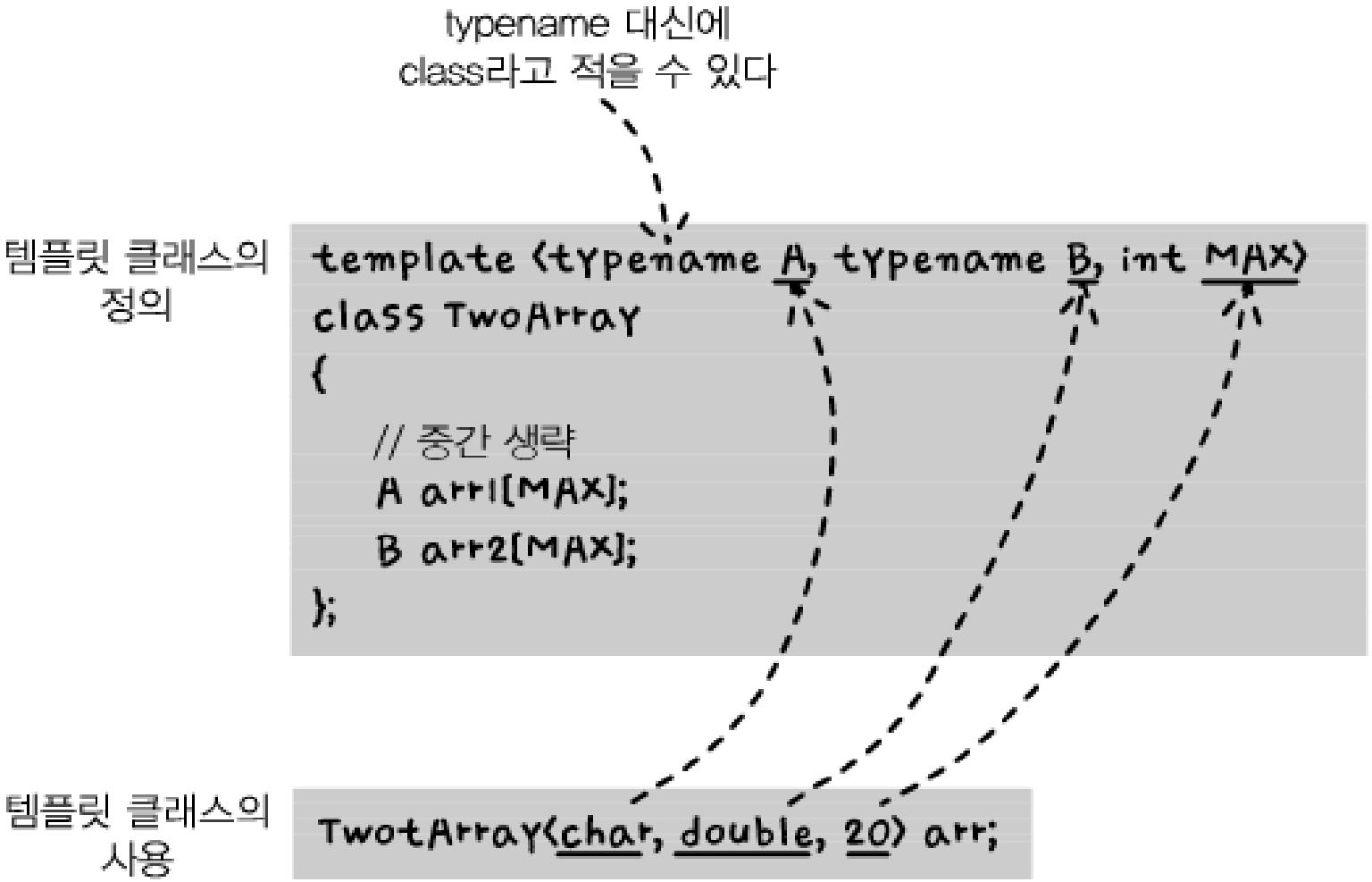
➔ AutoArray는 int에만 동작!

```
template <typename T>
class AutoArray
{
public:
      AutoArray(T* ptr)
      {   _ptr = ptr;  }
      ~AutoArray()
      {   delete[] _ptr;}
      T& operator[] (int index)
      {   return _ptr[index];     }
private:
      T* _ptr;
};

int main()
{
      AutoArray<float> arr( new float [100] );
      arr[0] = 99.99f;

      return 0;
}
```

➔ AutoArray는 모든 type에 동작!

Template과 STL

- Template 매개 변수의 사용



Template과 STL

- Template class의 이해
  - Template class 객체를 생성하는 순간 compiler 내부적으로 알맞은 class 생성
  - 개발자가 만든 code → complier가 새로 만든 class

```
template<typename A, typename B, int MAX>
class TwoArray
{
        // 중간 생략
        A arr1[MAX];
        B arr2[MAX];
};

TwoArray<char, double, 20> arr;
```

```
// 임의로 만든 생성된 class
class TwoArray_char_double_20
{
    // 중간 생략
    char arr1[20];
    double arr2[20];
};
```

Template과 STL

- Standard Template Library(STL): C++ template
  - Class, function 등의 집합체로 C++로 구현되어 있음
  - Array 및 pointer를 대체하는 data 관리에 용이

- STL 구성
  - Container
    - 같은 type의 원소를 관리하는 data 구조 지원 class
    - list, vector, deque, …
  - Algorithm
    - Container에 대해 복사, 전환, 병합, 정렬
    - random_shuffle, replace, fill, remove, sort, …
  - Iterator
    - 원소의 관리 방법 (pointer)
  - 함수 개체
    - 함수 연산자 ( )를 overloading

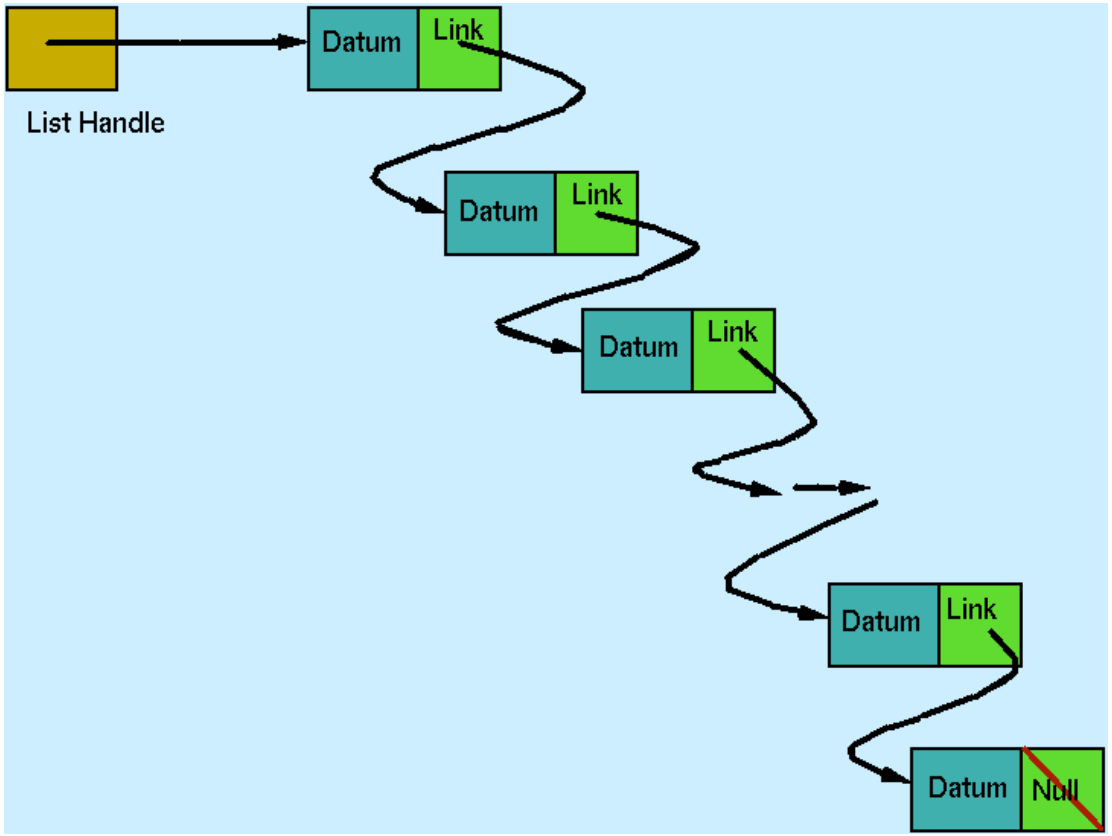
Template과 STL

- 자주 사용하는 STL: Container class
  - 같은 type의 원소(element)를 관리하는 data 구조 지원 class
  - 사용시: `#include <afxtempl.h>`

클래스	요약
vector	동적인 배열. 동적으로 원소의 개수를 조절할 수 있는 배열이다.
list	링크드 리스트.
deque	배열과 링크드 리스트의 장점을 모아놓은 컨테이너. 배열만큼 원소에 접근하는 시간이 빠른 동시에, 맨 앞과 끝에 원소를 추가하고 제거하는 시간에 링크드 리스트 만큼 빠르다.
map	맵은 원소를 가리키는 인덱스까지도 다양한 타입을 사용할 수 있다. 예를 들어서 다음과 같이 문자열 타입의 인덱스를 사용할 수도 있다. map<string, string> m; m[“add”] = “더하다.”;

Template과 STL

- Container class: Linked list
  - Data의 추가 및 삭제가 용의한 data 구조
  - Data의 접근(access)이 순차적(sequential)으로만 가능



CList

- MFC에서 제공하는 Linked list template class

```
// 정의 방법
CList <datatype> a;

// Data 추가
CList::AddTail(datatype newElement)
CList::AddHead(datatype newElement)

// Data 삭제
CList::RemoveTail();
CList::RemoveHead();
CList::RemoveAt(POSITION pos);
```

iterator

```
// 사용 예
CList <int> a;

// Data 추가
CList::AddTail(3)
CList::AddHead(4)
CList::AddHead(5)

// Data 삭제
CList::RemoveTail();
CList::RemoveHead();
POSITION pos = a.GetHeadPosition();
CList::RemoveAt(pos)
```

CList

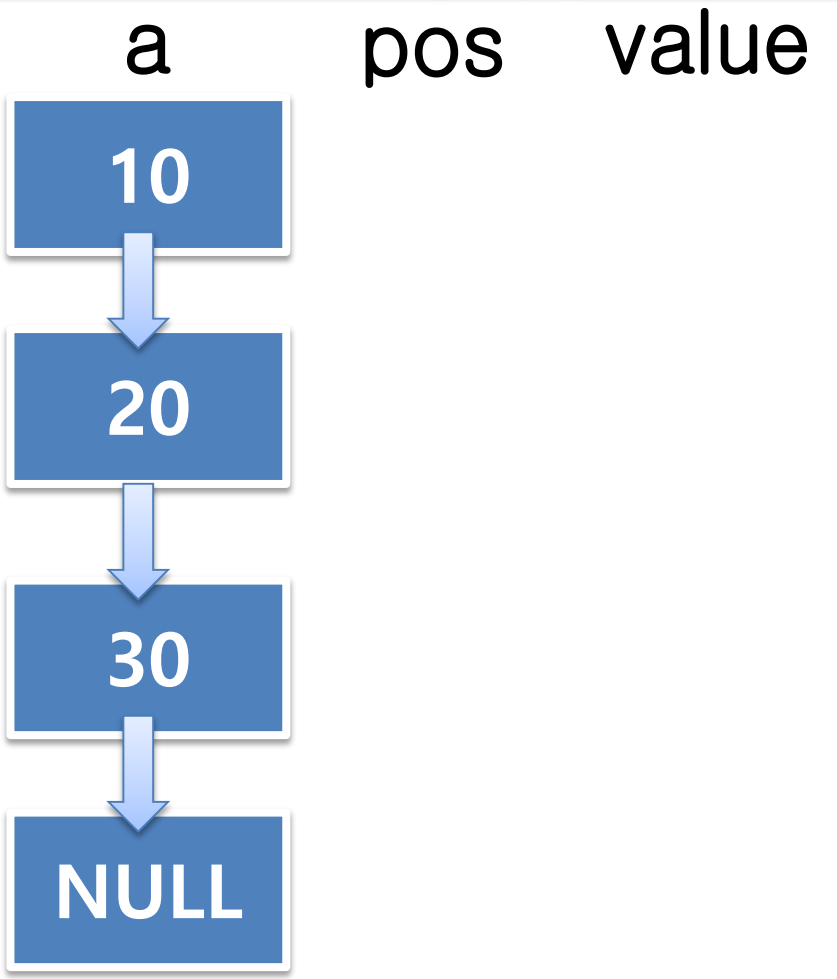
- Iterator
  - Container에서 pointer와 같은 역할을 하는 것
  - MFC의 iterator 변수형: POSITION
  - 사용 예

```
#include <afxtempl.h>

CList <int> a;

a.AddTail(10);
a.AddTail(20);
a.AddTail(30);

POSITION pos = a.GetHeadPosition();  // 첫 data 위치 반환

while(pos != NULL)
{
   int value = a.GetNext(pos);       // pos 위치의 data 반환 후,
                                     // pos는 list에서 다음 위치를 가르킴
}
```

CList

- Iterator 사용 예

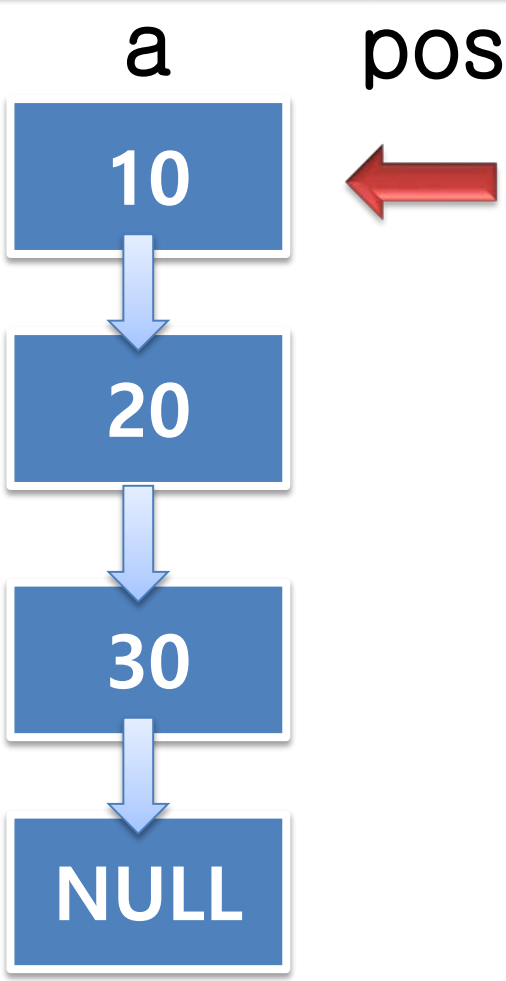
```
CList <int> a;
a.AddTail(10);
a.AddTail(20);
a.AddTail(30);

POSITION pos = a.GetHeadPosition();

while(pos != NULL)
{
        int value = a.GetNext(pos);
}
```



CList

- Iterator 사용 예

```
CList <int> a;
a.AddTail(10);
a.AddTail(20);
a.AddTail(30);

POSITION pos = a.GetHeadPosition();

while(pos != NULL)
{
        int value = a.GetNext(pos);
}
```

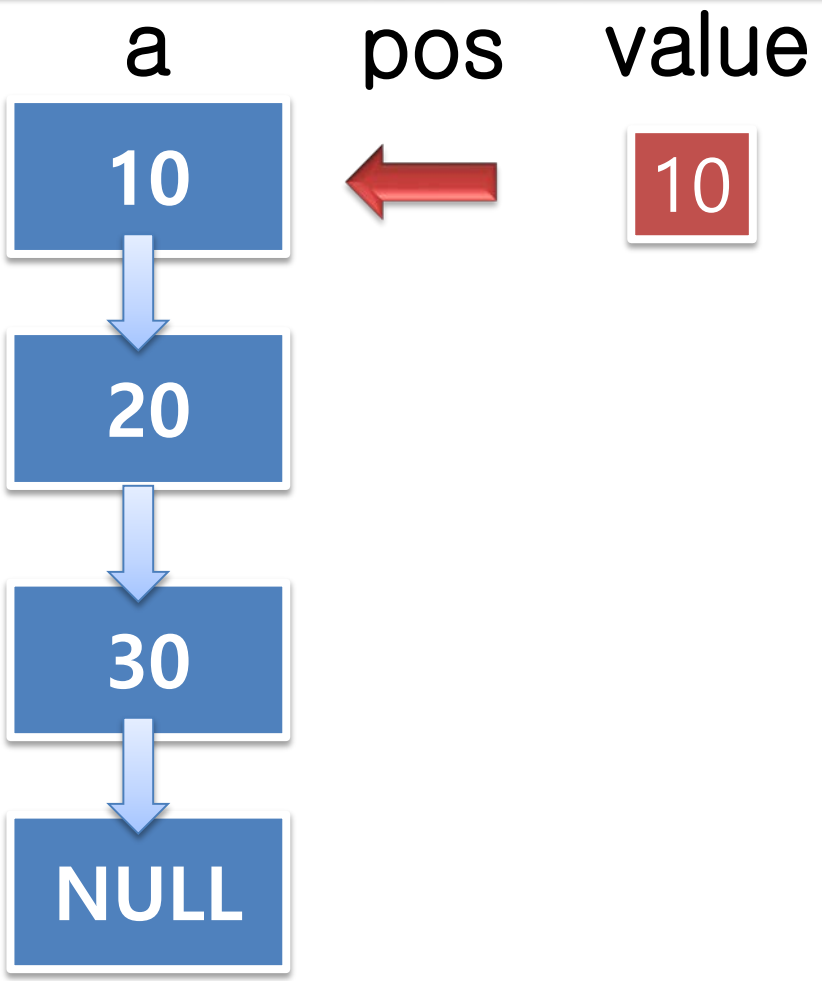


CList

- Iterator 사용 예

```
CList <int> a;
a.AddTail(10);
a.AddTail(20);
a.AddTail(30);

POSITION pos = a.GetHeadPosition();

while(pos != NULL)
{
        int value = a.GetNext(pos);
}
```

➔ 1회 호출

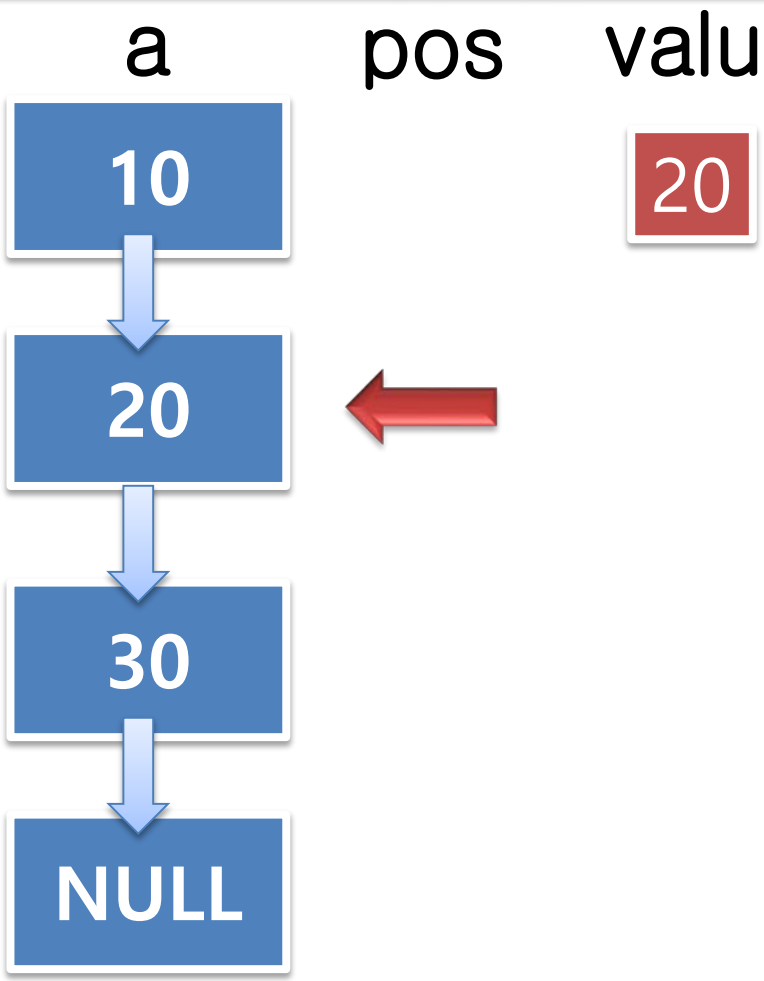


CList

- Iterator 사용 예

```
CList <int> a;
a.AddTail(10);
a.AddTail(20);
a.AddTail(30);

POSITION pos = a.GetHeadPosition();

while(pos != NULL)
{
        int value = a.GetNext(pos);
}
```

➔ 2회 호출

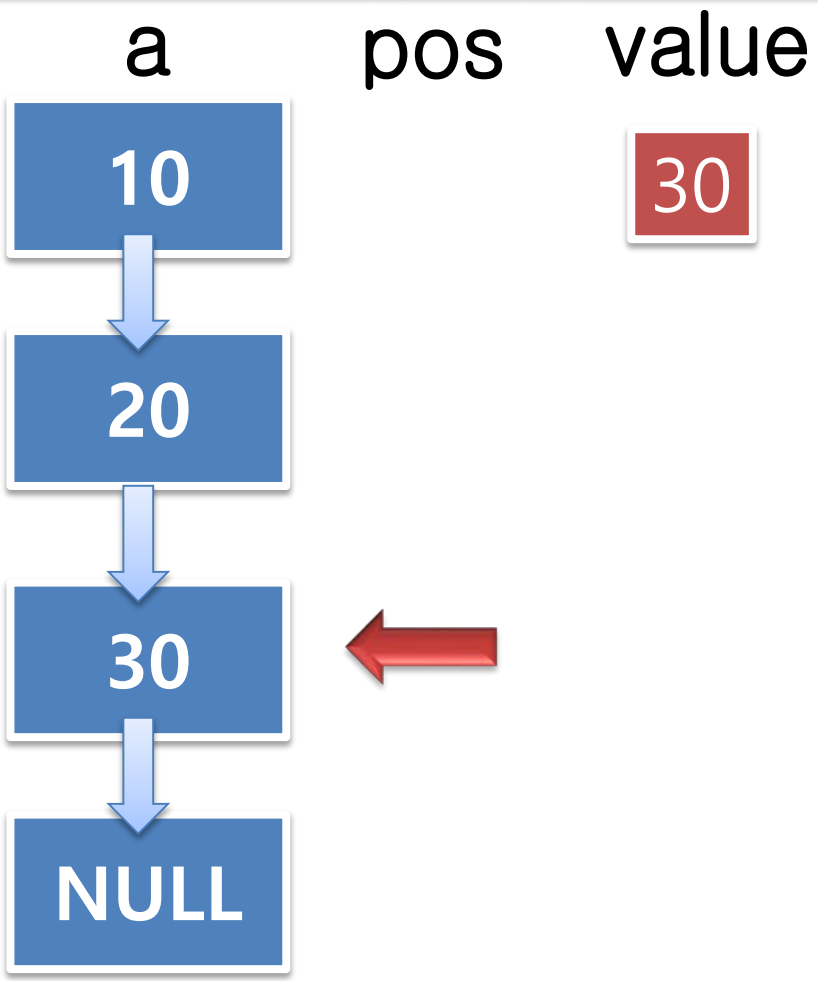


CList

- Iterator 사용 예

```
CList <int> a;
a.AddTail(10);
a.AddTail(20);
a.AddTail(30);

POSITION pos = a.GetHeadPosition();

while(pos != NULL)
{
        int value = a.GetNext(pos);
}
```

➔ 3회 호출

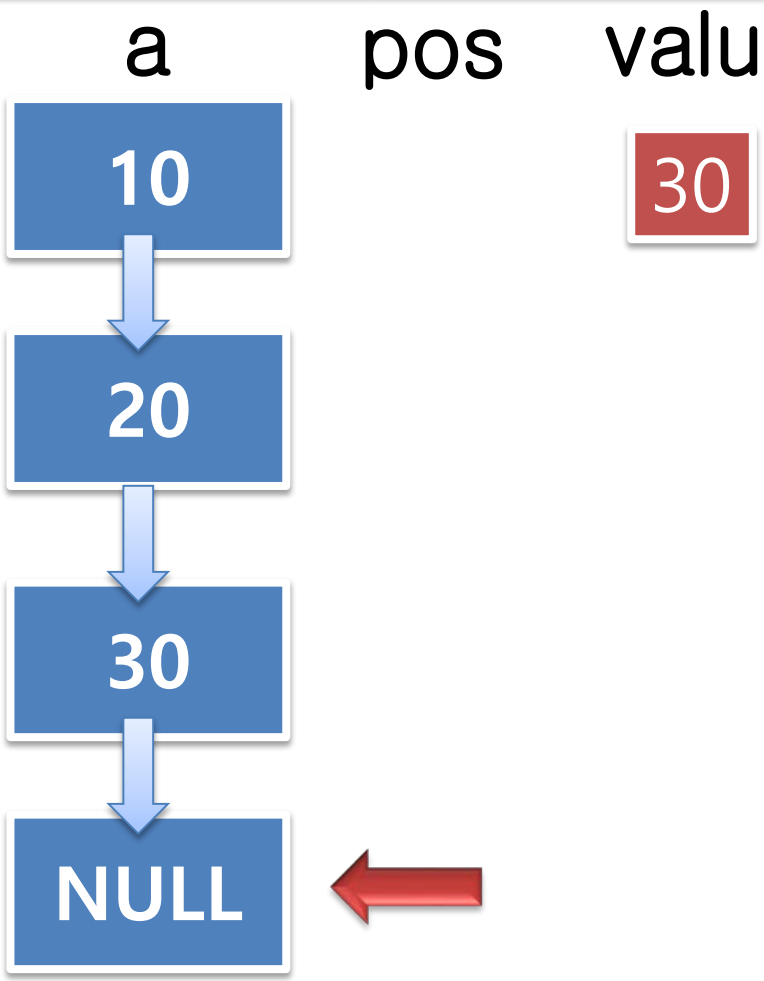


CList

- Iterator 사용 예

```
CList <int> a;
a.AddTail(10);
a.AddTail(20);
a.AddTail(30);

POSITION pos = a.GetHeadPosition();

while(pos != NULL)
{
        int value = a.GetNext(pos);
}
```

➔ 종료



CList

- CList의 구성 요소(element) 추가(insertion)

```
// 맨 앞에 추가
CList::AddHead(value)

// 맨 뒤에 추가
CList::AddTail(value)

// 임의의 위치에 추가
CList::InsertAfter(POSITION, value)
```

- CList의 구성 요소(element) 삭제(removal)

```
// 맨 앞 삭제
CList::RemoveHead( )

// 맨 뒤 삭제
CList::RemoveTail( )

// 임의의 위치 삭제
CList::RemoveAt(POSITION)

// 모두 삭제
CList::RemoveAll( )
```

CList

- CList의 구성 요소(element) 회수(retrieval)

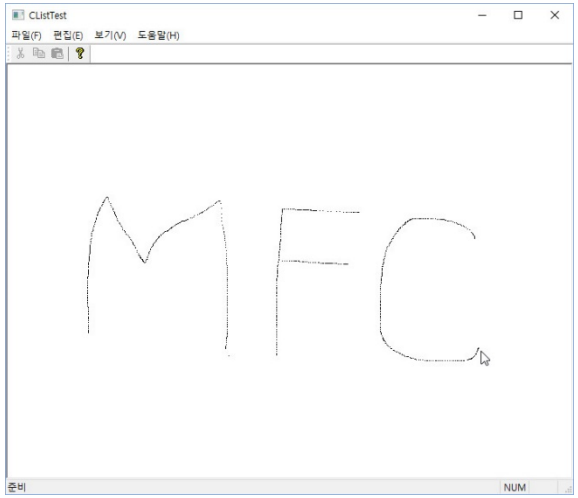
```
// 값 얻어오기
value  = CList::GetAt(POSITION)

// 값의 reference 얻어오기
value & = CList::GetAt(POSITION)
```

- CList의 구성 요소(element) 수정(modification)

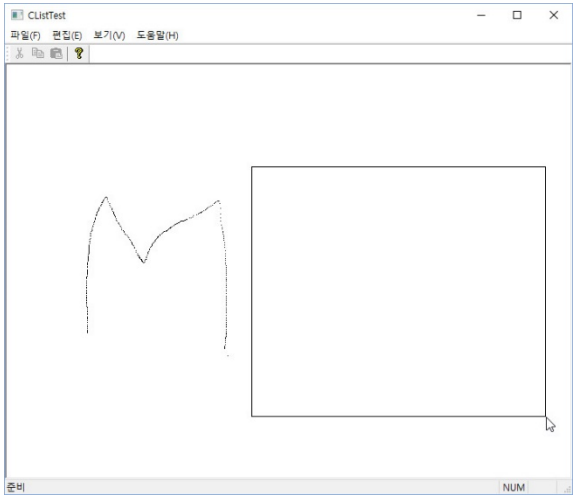
```
CList <int> a;

a.AddHead(10);
a.AddHead(20);
a.AddHead(30);

POSITION pos;
pos = a.GetHeadPosition();

int b = a.GetAt(pos);        // value
int &c = a.GetAt(pos);       // reference
```

CList

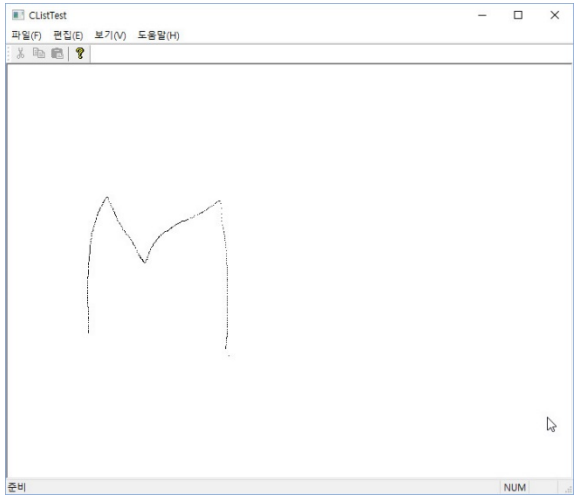
- 사용 예: 점(point) 찍고 지우기
  - 마우스 왼쪽 드래깅을 하면 그 위치에 점을 그리고, 점들의 위치를 CList를 이용하여 저장
  - 마우스 오른쪽 드래깅으로 사각형 안에 포함된 점들을 삭제



마우스 왼쪽
드래깅으로 점찍기



마우스 오른쪽
드래깅으로 사각형 지정



포함된 점들 지우기

Q & A